Checklist

The following checklist should be used when performing the individual inspection. Note that there is no universal checklist available. This one contains many common items, but is adapted and limited to the case at hand. Things missing are for example IO and exceptions.

Class Definition

1. Does each class have a descriptive name used in accord with naming conventions?
2. For each class and interface: Are the declarations ordered as follows: Class/interface documentation comment, class or interface statement, Class/interface implementation comment, Class (static) variables and constants, Instance variables, Constructors, Methods?
3. Does each class have an appropriate constructor?
4. For each member of every class: Could access to the member be further restricted?
5. Do any derived classes have common members that should be in the base class?
6. Can the class inheritance hierarchy be simplified?

Method

7. Are descriptive method names used in accord with naming conventions?
8. Is every method parameter value checked before being used?
9. For every method: Does it return the correct value at every method return point?
10. Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?
11. Do the values in units agree (e.g., euro vs. cents)?
12. Are there static methods that should be non-static or vice-versa?

Declarations

13. Are descriptive variable and constant names used in accord with naming conventions?
14. Are there variables with confusingly similar names?
15. Is every variable and attribute correctly typed?
16. Is every variable and attribute properly initialized?
17. Could any non-local variables be made local?
18. Are there literal constants that should be named constants?
19. Are there variables that should be constants?

Computation & Comparison

20. Is overflow or underflow possible during a computation?
21. For each expression with more than one operator: Are the assumptions about order of evaluation and precedence correct?
22. Are parentheses used to avoid ambiguity?
23. Are the comparison operators correct?
24. Is each boolean expression correct?
25. Are there improper and unnoticed side-effects of a comparison?

Control Flow

26. For each loop: Is the best choice of looping constructs used?
27. Will all loops terminate?
28. When there are multiple exits from a loop, is each exit necessary and handled properly?
29. Does each switch statement have a default case?
30. Are missing switch case break statements correct and marked with a comment?
31. Is the nesting of loops and branches too deep, and is it correct?
32. Can any nested if statements be converted into a switch statement?
33. Are null bodied control structures correct and marked with braces or comments?
34. Does every method terminate?

Performance

35. Can better data structures or more efficient algorithms be used?
36. Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?
37. Can the cost of recomputing a value be reduced by computing it once and storing the results?
38. Is every result that is computed and stored actually used?
39. Can a computation be moved outside a loop?
40. Are there tests within a loop that do not need to be done?
41. Can a short loop be unrolled?
42. Are there two loops operating on the same data that can be combined into one?

Comments

43. Does every method and class have an appropriate header comment?
44. Does every variable and constant declaration have a comment?
45. Is the underlying behavior of each method and class expressed in plain language?
46. Is the header comment for each method and class consistent with the behavior of the method or class?
47. Is only specified functionality implemented with no additional functionality added?
48. Do the comments and code agree?

49. Do the comments help in understanding the code?
50. Are there enough comments in the code
51. Are there too many comments in the code?

Packaging

52. For each file: Does it contain only one class?
53. For each file: Does it have the following ordering: Package and Import statements, beginning comments, Class and Interface declaration?
54. Does each line contain at most one statement?
55. For each line: Is it no more than about 80 characters long?
56. For each method: Is it no more than about 60 lines long?
57. For each class: Is no more than 2000 lines long?

Modularity

58. Is there a low level of coupling between packages and/or classes?
59. Is there a high level of cohesion within each package?
60. Is there duplicate code that could be replaced by a call to a method that provides the behavior of the duplicate code?
61. Are framework classes used where and when appropriate?